

ANEXO A
MANUAL TÉCNICO

A.1 Estructura principal

- ESP: Este apartado es el más pequeño del proyecto posee todo lo necesario para el arranque del sistema de sonido, posee un archivo “.ino” que contiene toda la lógica que será leída por el ESP para su arranque. También se posee un archivo de texto que contiene una breve explicación de su configuración y el arranque del mismo
- ESP-BACK: Actúa como el punto de entrada de datos en el sistema completo de monitoreo de ruido. Los datos capturados por este componente alimentan todo el pipeline de análisis, visualización y generación de reportes que se ejecuta en los componentes
- TFG-BACKEND: Contiene el servidor backend del sistema de monitoreo de ruido, desarrollado con Node.js y TypeScript. Este componente actúa como el núcleo central del sistema, proporcionando una API REST robusta que gestiona la recepción, procesamiento, almacenamiento y distribución de datos provenientes del ESP.
- TFG-FRONTEND: Contiene la aplicación web del sistema de monitoreo de ruido, desarrollada con React y TypeScript. Este componente proporciona la interfaz de usuario completa del sistema, ofreciendo un dashboard interactivo y responsive que permite a los usuarios visualizar, analizar y gestionar los datos a tiempo real.

A.2 ESP estructura

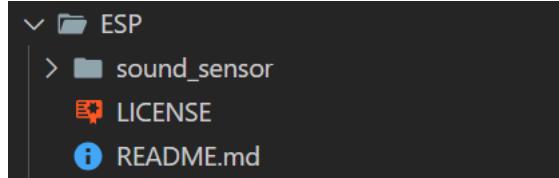


Figura A.1 Estructura del ESP

- El Readme es un corto pero explicativo documento de texto, donde se detalla la configuración de red del ESP y cuales son las variables modificables para su adaptación a la red de la universidad y su ubicación en los laboratorios para su uso.
- LICENSE es una licencia de uso MIT que define los términos legales bajo los cuales se distribuye el software.
- La carpeta "sound sensor" contiene un único archivo que será usado para el arranque del dispositivo ESP, este archivo .ino se correrá en un dispositivo ESP que será el encargado de generar reportes sobre la cantidad de sonido generado en los laboratorios, si el sonido excede un umbral este encenderá una alarma encargada de notificar a los encargados de donde proviene el sonido.

A.3 TFG-BACKEND

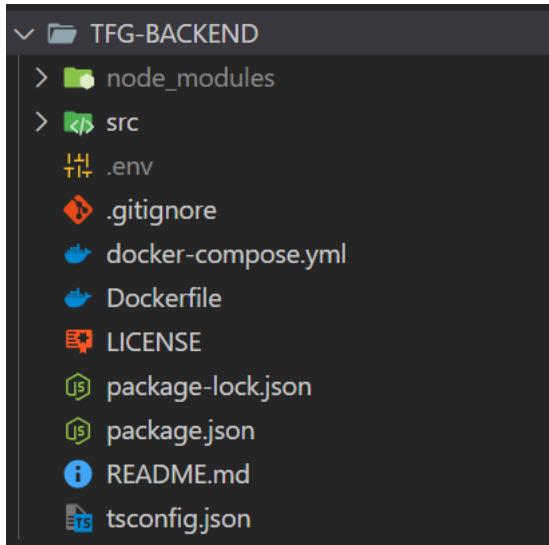


Figura A.2 Estructura TFG BACKEND

- Esta es la capa encargada del procesamiento de los datos que el usuario solicita desde la aplicación web

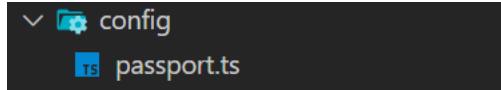


Figura A.3 ESP Carpeta Config

- Config: la carpeta config contiene la configuración central del proyecto. Aquí se definen y organizan parámetros importantes como la autenticación, las conexiones a servicios externos, y otras opciones necesarias para que la aplicación funcione correctamente y de forma segura.

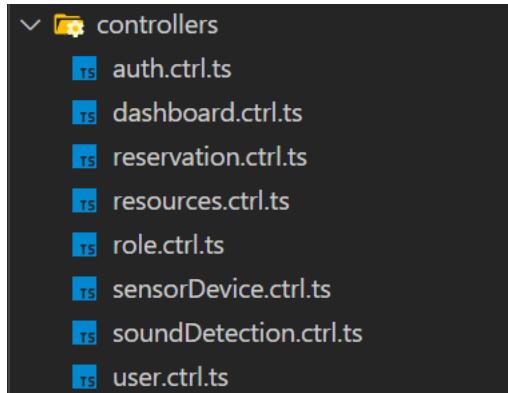


Figura A.4 ESP Carpeta Controllers

- Controller: la carpeta controllers contiene la lógica que recibe las solicitudes HTTP, coordina

la validación de datos y llama a los servicios correspondientes para manejar usuarios, autenticación, roles, sensores, detección de sonido, dashboard, recursos y reservas. Su función principal es actuar como intermediario entre las rutas de la API y la lógica de negocio.

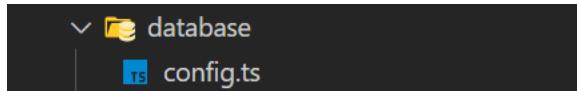


Figura A.5 ESP Carpeta Database

- **Database:** La carpeta database contiene el archivo que implementa la configuración y la conexión a la base de datos MongoDB utilizando Mongoose.

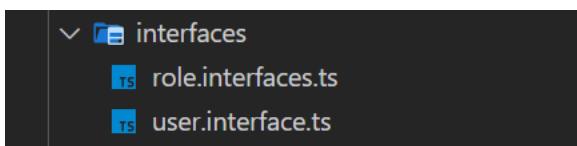


Figura A.6 ESP Carpeta Interfaces

- **Interfaces:** La carpeta contiene definiciones de interfaces TypeScript que describen la estructura de los datos utilizados en la aplicación, como los atributos requeridos para roles y usuarios, incluyendo campos como identificadores, nombres, descripciones, correos electrónicos y contraseñas. Estas interfaces ayudan a tipar y validar los objetos que se manejan en el backend.

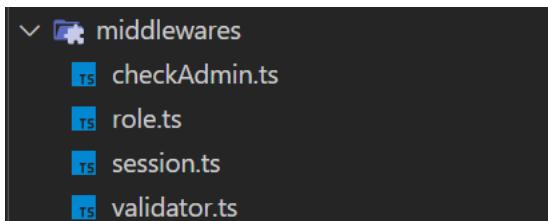


Figura A.7 ESP Carpeta Middlewares

- **Middlewares:** La carpeta contiene funciones intermedias que se utilizan para gestionar la autorización de usuarios, validar roles, manejar sesiones y verificar datos antes de que lleguen a las rutas principales de la aplicación. Estas funciones ayudan a controlar el acceso y asegurar que las solicitudes cumplan con los requisitos necesarios

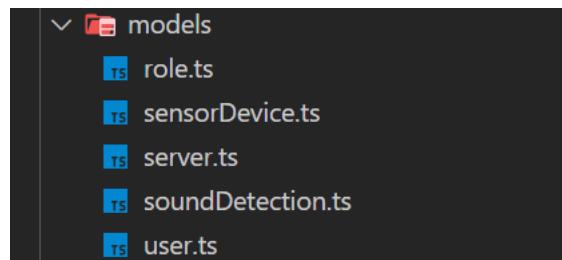


Figura A.8 ESP Carpeta Models

- **Models:** La carpeta contiene definiciones de modelos de datos utilizando esquemas. Estos modelos estructuran y gestionan como se almacenan y consultan los datos en la base de datos de la aplicación

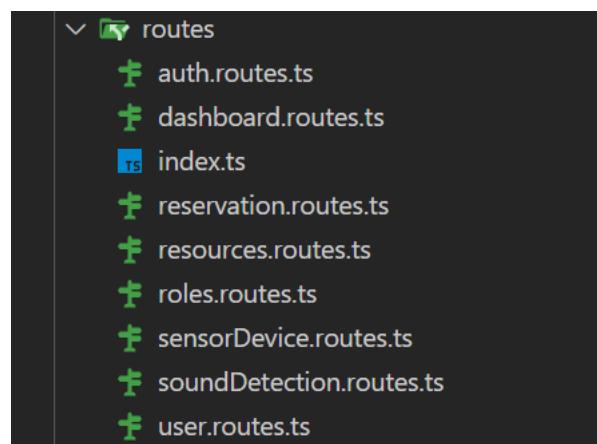


Figura A.9 ESP Carpeta routes

- **Routes:** La carpeta contiene archivos que definen las rutas de la API para gestionar la autenticación, panel de control, reservas, recursos, roles, dispositivos sensores, detección de sonido y usuarios. Cada archivo organiza los endpoints y controla como se manejan las solicitudes y respuestas para cada funcionalidad de la aplicación.



Figura A.10 ESP Carpeta Seeds

- **Seeds:** La carpeta contiene un archivo encargado de poblar la base de datos con datos iniciales o de prueba, facilitando la configuración y el desarrollo de la aplicación al crear registros pre-determinados necesarios para su funcionamiento.

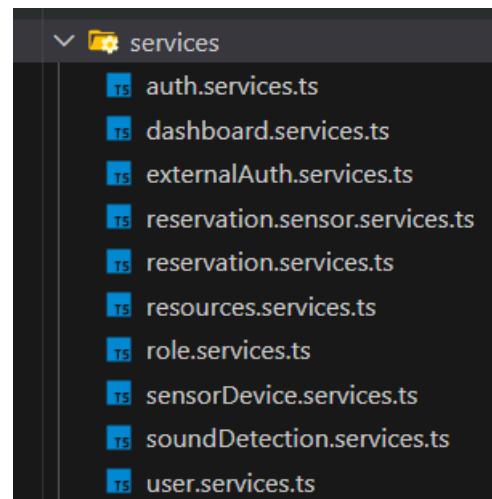


Figura A.11 ESP Carpeta Services

- Services: La carpeta contiene archivos que implementan la lógica de negocio para distintas funcionalidades de la aplicación, como autenticación, gestión de usuarios, roles, reservas, recursos, dispositivos sensores y detección de sonido. Estas funciones procesan los datos y coordinan las operaciones entre los modelos y las rutas.

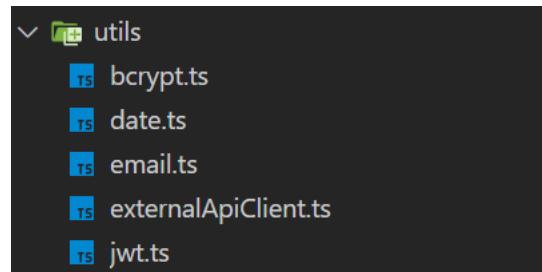


Figura A.12 ESP Carpeta Utils

- Utils: La carpeta contiene utilidades y funciones auxiliares para tareas comunes, como el manejo de contraseñas, fechas, envío de correos electrónicos, interacción con APIs externas y gestión de tokens JWT. Estas herramientas apoyan el funcionamiento general de la aplicación facilitando operaciones repetitivas o especializadas.

A.4 TFG-FRONTEND

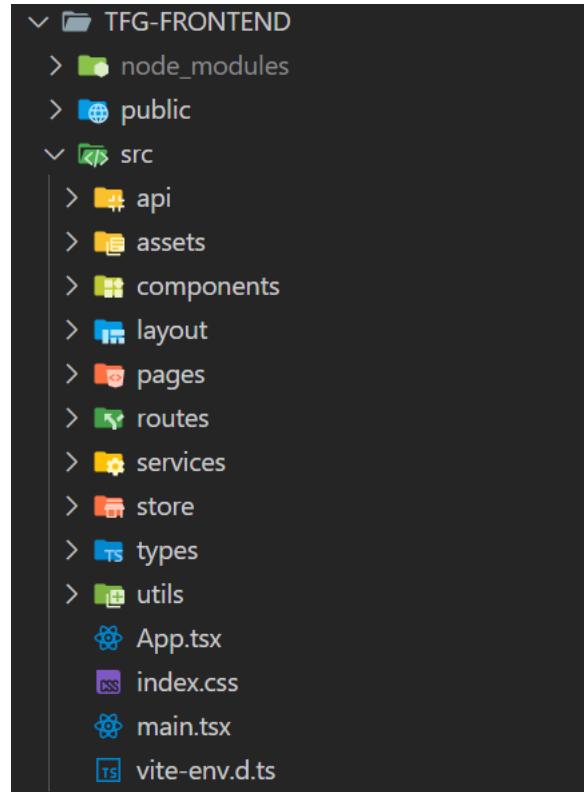


Figura A.13 ESP TFG FRONTEND

La carpeta TFG-FRONTEND contiene toda la implementación de la parte visual del sistema, la cual se desarrolló mediante Vite.js con React.js y Typescript.

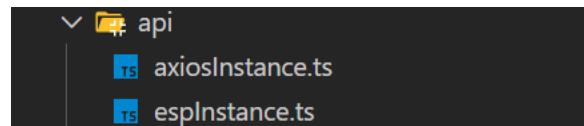


Figura A.14 Carpeta api

- Api: Esta carpeta contiene las configuraciones para centralizar las peticiones a la API así evitamos que los componentes hagan peticiones directamente.

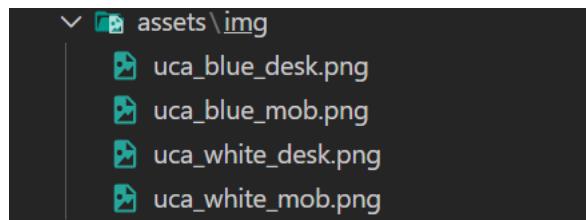


Figura A.15 Carpeta assets

- Assets: La carpeta assets contiene los recursos estaticos que se utilizaron en los diferentes componentes, como son las imagenes, iconos y archivos SVG.

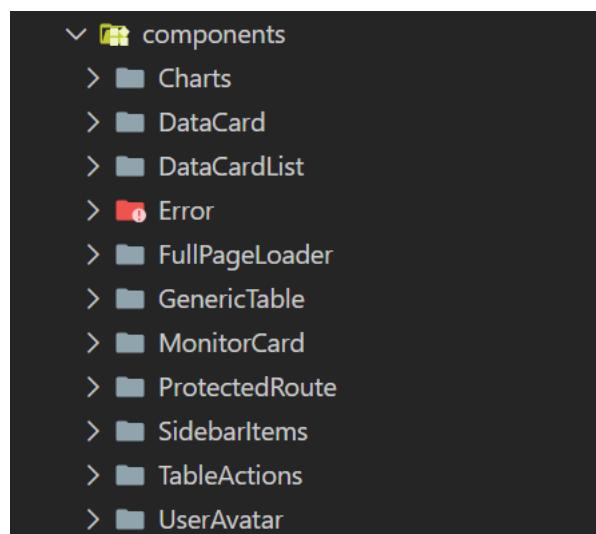


Figura A.16 Carpeta components

- Components: Esta carpeta contiene todos los componentes reutilizables y desacoplados para solo utilizarlos cuando sean necesarios, por ejemplo, botones, cards, inputs, entre otros.



Figura A.17 Carpeta Layout

- Layout: En la carpeta layout tenemos las estructuras generales de la aplicación, aqui se define la forma del estilo base de la página.

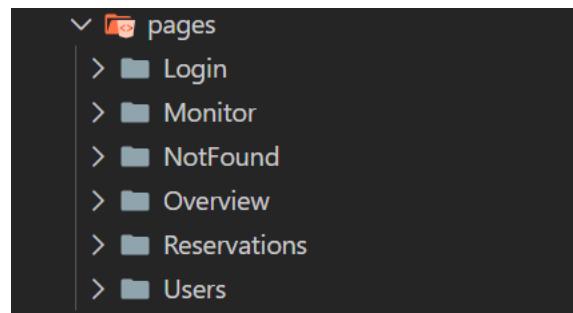


Figura A.18 Carpeta Pages

- Pages: En esta carpeta tenemos todas las vistas de nuestra aplicación, representan rutas complejas y cada una tiene su propia lógica.

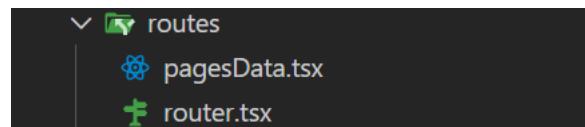


Figura A.19 Carpeta Routes

- Routes: La carpeta routes contiene definidas todas las rutas de navegación

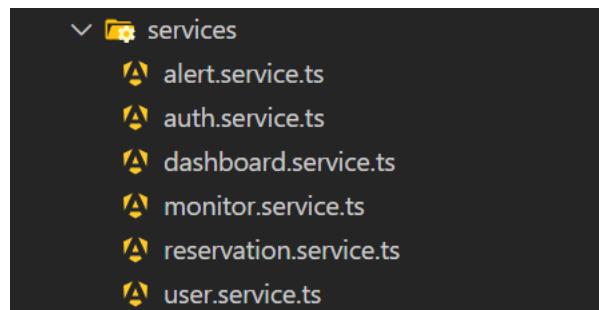


Figura A.20 Carpeta Services

- Services: En la carpeta services tenemos toda la lógica de negocio de nuestro sistema, como los servicios de autenticación, manejo de usuarios, entre otros.

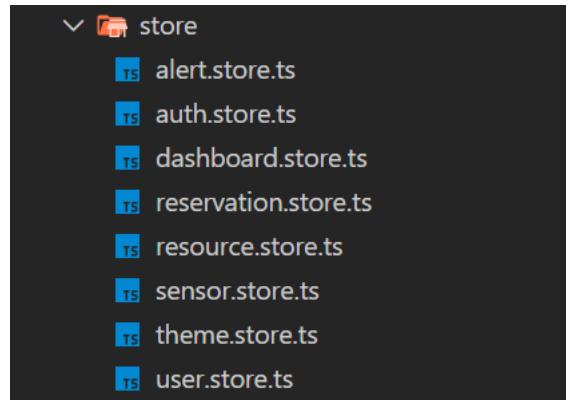


Figura A.21 Carpeta Store

- Store: La carpeta store contiene el estado global de la aplicación, cada archivo .store.ts contiene un store independiente encargado de gestionar un dominio funcional específico, como autenticación, usuarios, reservas, sensores o temas visuales.

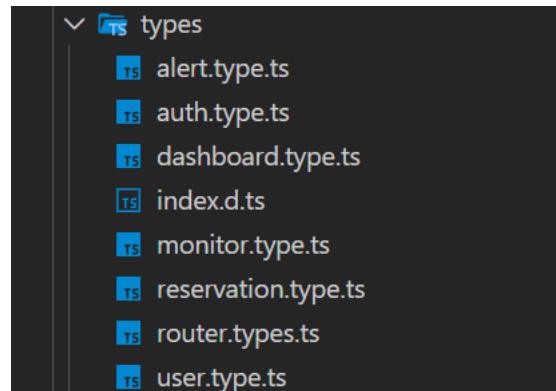


Figura A.22 Carpeta Types

- Types: La carpeta types contiene definidos todos los tipos o interfaces usados en la aplicación

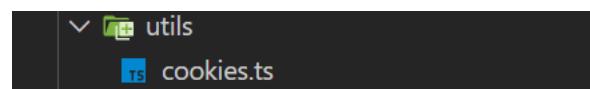


Figura A.23 Carpeta Utils

- Utils: En la carpeta utils tenemos funciones para la gestión de las cookies en nuestro proyecto.